Automatic Extraction of Go Game Positions from Images: A Multi-Strategical Approach to Constrained Multi-Object Recognition

Alexander K. Seewald

Seewald Solutions

1180 Vienna, Austria

alex@seewald.at

December 19, 2009

Abstract

Here, we present a constrained object recognition task that has been robustly solved largely with simple machine learning methods, using a small corpus of about 100 images taken under a variety of lighting conditions. The task was to analyze images from a hand-held mobile phone camera showing an endgame position for the Japanese board game Go. The presented system would already be sufficient to reconstruct the full Go game record from a video record of the game, and thus is complementary to (Seewald, 2003) which focuses on solving the same task using

different sensors. The presented system is robust to a variety of lighting conditions, works with cheap low-quality cameras and is resistant to changes in board or camera position without the need for any manual calibration.

Keywords: Object Recognition, Machine Learning, Computer Vision, Applied Research

1 Introduction

We present a multi-strategical approach to multi-object class detection under constraints, more specifically the recognition of board states of the Japanese game of Go from still images. Research in the field of Machine Vision has for example focused on generating useful features from images for purposes of object recognition (e.g. SIFT (Lowe, 2004), Radial Symmetry Transform (Loy & Zelinsky, 2003)), on image classification with a small set of classes (e.g. handwritten digit recognition with Convolutional Networks (LeCun et al., 1998) or Support Vector Machines (Seewald, 2005)), or on image similarity for purposes of searching (e.g. Wavelet Transform (Jacobs et al., 1995)). There are also algorithms for several standard tasks in image preprocessing, e.g. the Canny Edge Detector (Canny, 1986), the Harris Corner Detector (Harris & Stephens, 1988) and the Generalized Hough Transform for detecting arbitrary objects (Duda & Hart, 1972), which are widely used. Machine learning is also widely used within this research field, but conventional methods focus on using a single learning system. Here, we focus on a multi-strategy approach, which combines several learning

systems that are applied in sequence, using diverse input data representations. While the system strongly relies on already known object recognition schemes using SIFT keypoint descriptors (used e.g. in (Mikolajczyk, Leibe & Schiele, 2006)), Canny Edge Detector algorithm and pixel-based representations, and uses logistic regression and Support Vector Machines (SVM) as learning systems, we believe that the combination as sequential ensemble is novel. The board recognition algorithm from Section 4.3 is clearly novel as it is based on ad-hoc programming, but constitutes the weakest link in the system. Lastly, the achieved performance is quite good and constitutes both the first quantitative result reported for this specific task as well as the first with sufficient performance to be applied in practice. One major reason for the observed performance is that we utilized the regular board structure to filter the output of the object recognition modules and thus simplified the task beyond a usual multi-object recognition task where such structure is usually not present.

The task of extracting arbitrary information from images in a general setting can be greatly simplified by controlling lighting conditions and camera position. Just calibrating a given image processing system to a new uncontrolled environment may take several hours and can be a painstaking manual process.[1]

In order to improve on the state-of-the-art by building more robust systems that work well under a variety of lighting conditions, we have focused on a more challenging object recognition task. It should work without manual calibration

[1]Personal communication by Dr. Markus Würzl of the award-winning Robo-Cup team Austro, Institute for Handling Devices and Robotics, Vienna University of Technology.

and under uncontrolled lighting conditions. The camera position is different for each image, as the capture device is a hand-held mobile phone camera. No flash was available. The picture quality is mediocre at best and only offers 640x480 (0.3 megapixel) resolution. The game board is a common woodden Go board with white and black stones. For reasons which have to do with an earlier project, where we built an embedded device to record moves with a sensor array (Seewald, 2003), we used an 8x8 board instead of the more customary 9x9, 13x13 or 19x19 sizes. In this earlier project, the 8x8 size was forced on us due to hardware restrictions.

We will proceed to describe two variants of the system that are able to correctly recognize between two thirds and three quarters of still images showing Go board endgame positions without errors, and misclassify two to four of the 64 board positions per image in the remaining images, yielding an overall error per board position of between 0.8% and 2.03%. The faster worse-performing system can process one frame about every five seconds with some additional optimizations. We expect that this system would be able to recognize at least 98.39% of full game records from a real-time analysis of a game's video stream under reasonable assumptions.

2 Related Research

(Mikolajczyk, Leibe & Schiele, 2006) present an approach for simultaneous recognition and localization of multiple object classes and as such are similar to

our approach. However, while they use PCA and a tree structure, we use the full 128-dimensional SIFT keypoint vector and a SVM classifier. Our approach also differs in that we use the known structure of the board as a regular grid and thus constrain the position of stones, which greatly improves recognition performance.

(Scher, 2006) propose a system to determine the whole Go game record from analysis of the video stream from a fixed camera. Initially, board corners are input by the user and perspective correction is applied. Here, our approach differs as we recognize board corners automatically, separately for each image. Then, Sobel edge filter followed by Hough transform for circles was used to detect stones in the images. This was also the first approach we tried, but performance of Sobel was clearly inferior to parameter-optimized Canny. The author seems to agree, as this was called a *weak classifier* due to its high error rate. To prevent frequently occurring misclassification of hand and stone movements[2], optical flow analysis was applied to determine areas of strong movement, where the output of the stone detector was ignored. The Viterbi dynamic programming algorithm was applied to construct a likely sequence of moves, given the noisy outputs of stone and motion detector. This proved to reduce noise significantly, due to the high redundancy of the video signal. While the resulting system is probably not competitive to our approach in terms of accuracy (no quantitative results are given; tested only on a single video sequence), it would

[2]During capture in Go, a set of stones are taken from the board. Players will also occasionally adjust stone positions during normal play, thus touching their own and enemy stones for a short time period.

be feasible to apply our system to video analysis using a similar approach to improve performance further. It also opens up the intriguing possibility to combine our approach with sensor data from the original Intelligent Go Board project (Seewald, 2003).

(Deuk-Cheol et al., 2005) discuss a simple system to extract Baduk (Chinese for Go) game records from TV programs. They use horizontal and vertical histograms after application of a Roberts edge detector to find the board, and size/shape constraints (e.g. board must be square, minimum size, minimum distance from border) to drop video frames where the board is obscured by the players' hands. A simple iterative algorithm is then used to determine brightness intervals for black and white stones, and the brightness interval outside these areas is analyzed to determine horizontral and vertical lines within the board by periodicity. There is no quantitative evaluation – they just state it "works pleasingly" – but it seems that at least in some cases recognizing the board does not work well, which would be expected from these simple methods. They only show opening game records where such a approach works best due to the small number of stones on the board (which creates a clear line signal), so their approach might need some finetuning to work throughout a full game.

(Hirsimäki, 2005) describes a system to recognize Go board states from single images. It uses the Hough Transform for lines and some subtle filtering to determine playing field position, which only works if there are not too many stones on the central lines. This weakness makes it unsuitable for a typical endgame position, which is consistent with the observation that one of our

preliminary approaches using Hough transform for lines also failed there. He proposes a linear 5x5 filter to detect stones and lines, which incorporates domain knowledge and thus might be interesting to compare to our optimized Canny. There is no quantitative evaluation, but as the six shown images are of quite high quality and taken under good lighting conditions, we would be bound to expect that it does not quite reach the performance of our system.

(Shiba & Mori, 2004) describe a system that uses genetic algorithms to detect the contour of a Go-board and which they claim works reasonably well. However, they just evaluate this first step and do not propose a full system as we do here. Also, competitive faster alternatives such as Hough transform for lines were not considered.

(Ball, 2004) describes a system written in Perl, available from CPAN as `Games::Go::Image2SGF`, which also aims to recognize Go board states from single images. It uses a simple sampling approach to distinguish white, black and empty stones. However, the user has to specify the four corners of the board manually. After this a simple 3D model based on barrel distortion is applied, and at each grid point, a small circular area of pixels is sampled. The author claims that the simple sampling approach should suffice for stone detection, but this has not been tested comprehensively.[3] However, for the stated purpose of analyzing video streams of Go games, a simple approach might suffice, because the video signal has so much redundany, and additional constraints on permissible moves can be applied – similar to (Scher, 2006) above.

[3]Pers. comm. by the author.

(Lowe, 2004) describe the Scale Invariant Feature Transform (SIFT), which is a state-of-the-art keypoint descriptor. Given an image, it is sampled at multiple scales and robust keypoints are extracted, which are then described by scale, orientation and a 128-dimensional keypoint descriptor. We use the 128-dimensional keypoint descriptor to classify keypoints into one of several classes related to relevant board objects such as stones, crossings, and borders, and use scale and orientation for filtering stones as well as determining the local orientation of t-segments.

(Loy & Zelinsky, 2003) describe the Radial Symmetry Transform, which is a keypoint detector for radial symmetry. It is quite fast and could easily run in real-time on video data. The Radial Symmetry Transform would be much better suited than SIFT to recognize white and black stones, but would also be much less well suited to recognize crossings and corners which are by their nature not radially symmetric. Another disadvantage is that it does not return a keypoint descriptor which is likely to be necessary for further filtering, and might be more costly to compute than our approach of parameter-optimzed Canny plus local Hough transform for circles.

3 Data Collection

Here, we describe the data collection of training and test images, and show some real-life sample images from our corpus.

Initially, we utilized six images previously recorded under arbitrary lighting

conditions with arbitrary game positions, which had been used as testcase for the Intelligent Go Board project. These were the base for initial experiments on the feasibility of the whole task.

Afterwards we opted for a more systematic approach, and let the Computer Go player Gnu Go 3.6.[4] play against the author ten times with different randomization settings, afterwards recording the ten different endgame positions. Each of these was then a) used as is; b) rotated by 180 degrees; c) inverted (i.e. switching black and white stones); and d) rotated *and* inverted. This yielded four times ten representative endgame positions for our experiments. The reason for using endgame positions is that for these it is hardest to find the board position using the Hough transform for lines, which has been previously applied to this task. This approach also ensures a reasonably equal number of black and white stones and empty positions, whereas at the beginning the number of empty positions is much larger.

We recorded images under five different lighting conditions: bright daylight, daylight on a cloudy day, halogene lamps, halogene lamps plus cloudy daylight, and flourescent indirect lighting. For each board position, we recorded a hand-held photo from each player's side, yielding around 20 photos per lighting condition (either (a) & (b), or (c) & (d), as we randomly chose to use either only normal or only inverted endgames for each lighting condition). All photos from one lighting condition were recorded in one setting taking less than 20 minutes. Some photos had to be removed because the board was cut off or distorted too

[4]Available under `http://www.gnu.org/software/gnugo`

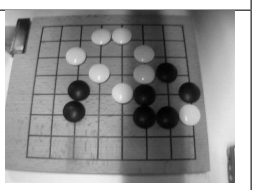
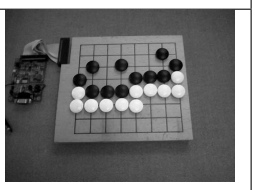
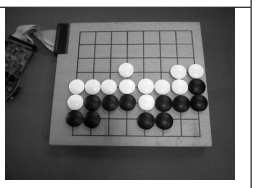
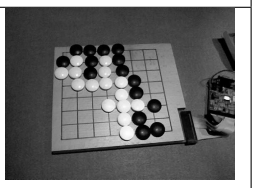
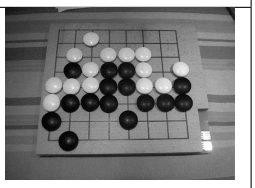
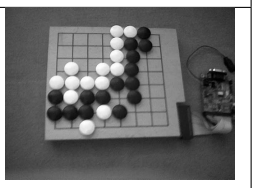
Lighting condition	Training Size	Test Size	Sample image
Initial (uncontrolled)	6	0	
Bright Daylight	10	10	
Cloudy Daylight	8	9	
Halogene Lamps	10	10	
Halogene Lamps plus Daylight	14	9	
Flourescent indirect	9	9	

Table 1: Training and test sets by lighting conditions.

much, so that some stones were no longer visible. Table 1 shows an overview of the training and test sets with sample images.

The images were randomly distributed into training and test sets. For Halogene Lamps plus Daylight, we biased this to ensure approximately the same number of test images as for the other lighting conditions. We manually marked the corners of each board and each board position by type (white stone, black stone, or crossing (empty) – see also Figure 2) and its exact position within the image. For keypoint classification, it proved essential that the position of each crossing, T-segment and corner is known as precisely as possible. Black and white stones proved to be less sensitive to exact positioning and keypoints further away could successfully be assigned to these classes. As is customary, the training data was used to optimize and finetune the system while the test data was used at the very end to get a reasonable estimate of the system's performance on unseen data. For reasons of data utilization efficiency, we did not use a separate validation set. Note that we did not give information about lighting conditions to the learning systems, or use it in any other way for optimizing the system. The rationale for this was to force the learning systems to build models independent of lighting conditions (in some cases simplified by using SIFT keypoint descriptors as input), and as we shall see later this seems to have worked.[5]

[5] In one small unsystematic experiment, we found that adding a field encoding the lighting condition to the training data roughly halved(!) the error rate of the learning system.

4 Final System

Our rationale for creating the system was to start with a reasonably simple learning task (such as multi-object classification using SIFT keypoint descriptors as input to a learning algorithm), and then continue to add complementary learning tasks until the desired performance was reached. Except for step 3., which still involves an ad-hoc board estimation algorithm, we have succeeded in this. The final system consists of the following steps, which are executed in sequence.

1. SIFT keypoint detection (Lowe, 2004) on a grayscale version of the original image. We also considered (Loy & Zelinsky, 2003), but it could not be used, because non-symmetric features such as crossings would not be recognized, and no keypoint descriptor is returned for their approach.

2. Applying the keypoint classification model on all SIFT keypoints. The model was learned from all training data.

3. Estimating board position (i.e. all four corners) by analyzing the structure of classified keypoints. Afterwards, a 3D model is fitted to the corner points to accurately estimate the position of each field on the board. Note that for Go, a field is centered on the crossing between two lines. See also Figure 2.

4. Stone detection via Canny edge detector and threshold on the proportion of border pixels in a circular area centered around stone position. Parameters for canny and threshold were estimated from training data. Stone

color (black, white) was computed via average brightness of circular area around expected center.

5. Stone and empty field detection via keypoint classification. Only applied for board positions not previously recognized.

6. Stone and empty field detection via model on average and standard deviation of brightness around the expected stone position. The model was learned from all training data and was also applied only to remaining board positions not previously recognized.

Note that steps 2., 4., 5. and 6. rely on machine learning algorithms which learned from the training data. No work needed to be done on 1. – we used the SIFT demo version as is – and thus 3. was the only step that involved significant ad-hoc programming by the author. The implementations of all learning algorithms are due to the open-source tool WEKA (Witten & Frank, 2005) and have been written in Java. Image processing was done via the open-source java library ImageJ (Rasband, 1997-2006) and additional code due to (Burger & Burge, 2006). A public domain Canny implementation by Mike Heath (`heath@csee.usf.edu`) written in C was also used. As we already mentioned, we used the freely available implementation of SIFT, which would need to be licensed for commercial applications.

Figure 1 shows each step in turn with a sample image. Left-to-right, top-to-bottom shows step 1 to 6: original image with keypoint as small white crosses; keypoints classified as T pieces (on the border, $\top$), corners ($\angle$), crossings(+)

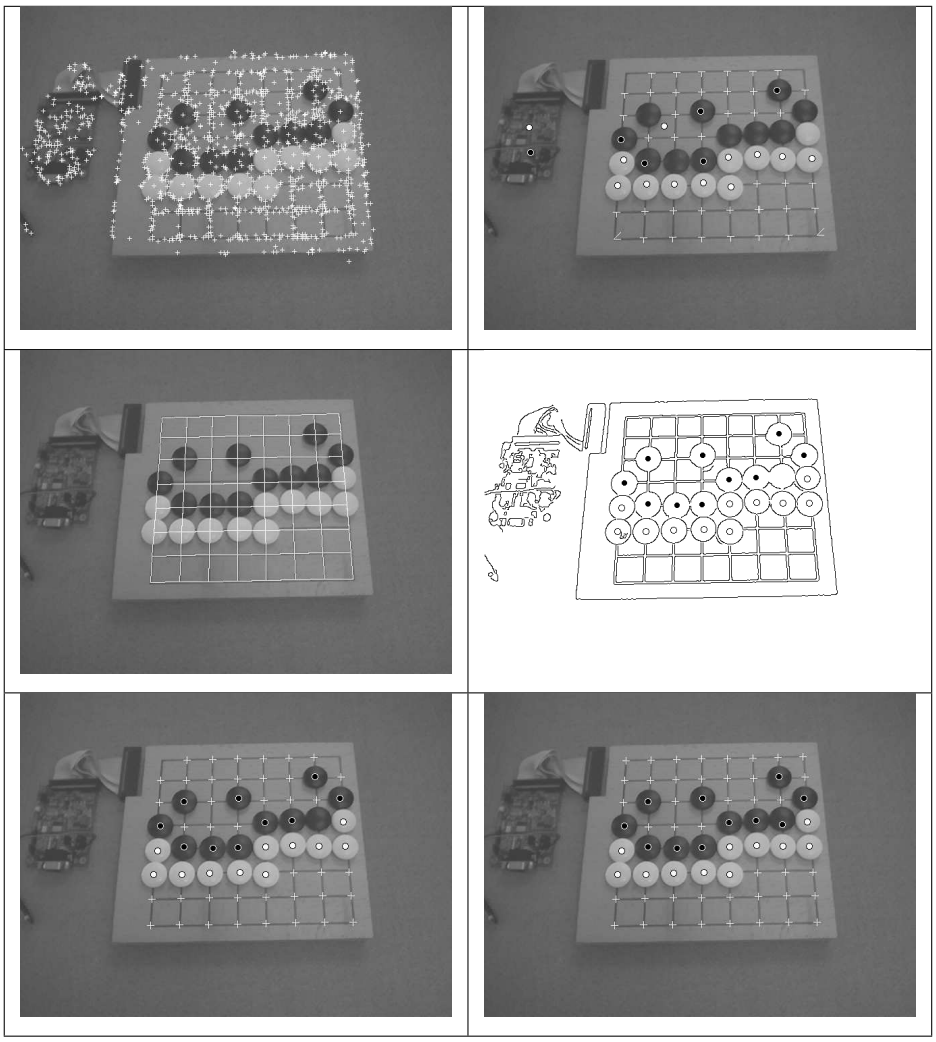


Figure 1: Steps 1-6 with sample images after each step, left-to-right, top-to-bottom.

and white/black stones (○/●); final board position after step 3; output of Canny edge detector and white/black stones; previous steps plus output from keypoint model; and final output after applying last ditch model. We will now go through each relevant step in turn. Section 4.X documents step X in detail.

4.1 SIFT Keypoint Detection

Each image was transformed into grayscale. We ran the demo version of SIFT from Lowe's home page at `http://www.cs.ubc.ca/~lowe/keypoints/`. Output was a file containing all found keypoints with their scale, orientation and 128 element numeric description vector.

4.2 Keypoint Classification

The first task was to find out where the board appears in the image. We soon found out that it is far from trivial to distinguish crossings (.), border pieces (T), corners (C), black stones (B) and white stones (W), and of course irrelevant keypoints (N) from one another. Figure 2 shows a sample image with examples for all keypoint types for clarity. See also top-left of Figure 1 for a real-life example.

(Lowe, 2004) describes an approach for object recognition that uses one prototype per class.[6] By computing the Sammon map (Sammon, 1969), which is a non-linear distance-preserving mapping from high-dimensional into low-dimensional spaces, we can see that this approach is not sufficient. The visualization of 128-dimensional keypoint space into two dimensions with stress=0.0814, see Figure 3, confirms that a more elaborate approach is needed to distinguish all the categories, as e.g. corners, T segments and crossings are overlapping to a large degree. Note that the irrelevant keypoints are not shown here for clarity, but are in the vast majority with 91.1% of all keypoints, thus creating a further

[6] This is equivalent to coarse instance-based learning with one example per class.

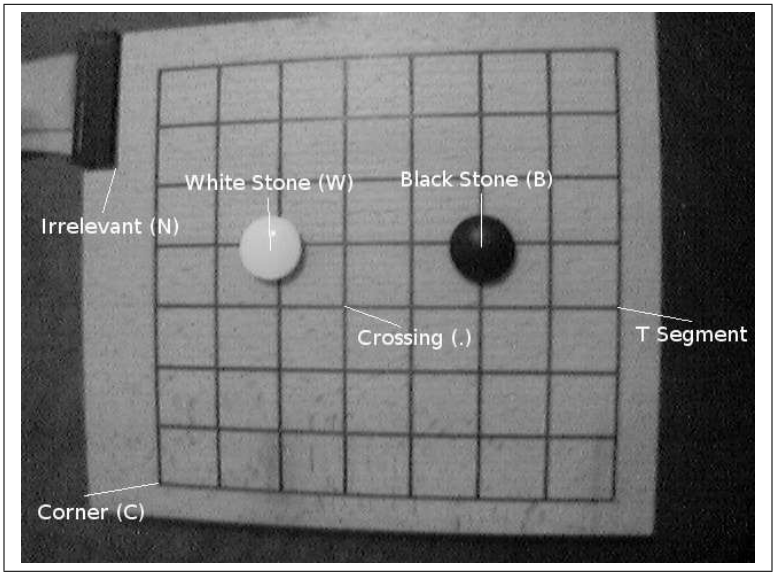


Figure 2: Sample image with keypoint types.

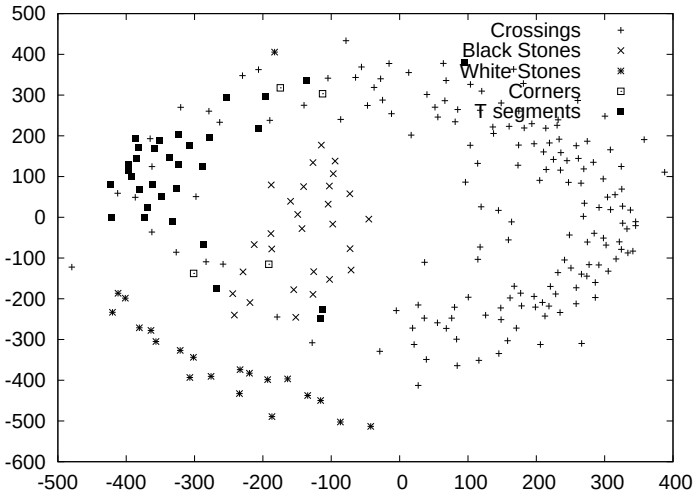


Figure 3: Sammon mapping of the keypoint space, stress=0.0814.

challenge.

Therefore, we used the full 128-element numeric keypoint description vector

from SIFT as input into a Support Vector Machine learning algorithm trained via Sequential Minimal Optimization (Platt, 1998) whose task was to predict the different categories. This worked very well already with a linear kernel, and the error rate could be approximately halved by parameter optimization.

Training data was automatically generated from the tagged training images and the output of SIFT. For corners, T segments and crossings, all keypoints within 1/16th of the field size (around 1-2 pixels) were considered relevant; and for black and white stones, all keypoints within 1/3rd of field size and with a scale of between 1/4th and 1.1 of field size were considered relevant. All other keypoints were considered to be irrelevant. Top-right of Figure 1 shows the results of applying the learned model on a real-life image.

4.3 Estimating Board Position

The keypoint model gives us a good but not perfect list of candidate corners, crossings and T segments. Since all four corners are needed, simply using the keypoints classified as corners is not sufficient, as there may be no keypoint corresponding to a specific corner (e.g. top right of Figure 1 misses two corners). We chose the more robust approach of utilizing the T segments, of which there are up to 6 per board side. Additionally, we found that the orientation of the border pieces according to SIFT was a very good approximation of the direction of the local border line, although it could be 180 degrees minus the orientation angle in some cases.

We began by considering each T segment separately as two candidate lines

with orientations of angle and 180 minus angle, respectively, and incrementally combined them with new T segments when they were less than 16 pixels apart. After each combination, the line was recomputed via linear regression on all constituent T segments.

Afterwards, we removed all lines which were less than 80 pixels from the center of the image, thus utilizing a reasonable constraint on minimum board size. Lines with two or more T segments were retained. Those lines with only one T segment were only retained if a corner, a crossing, a white or a black stone, was less than 32 pixels away from the line by normal distance, thus lending independent support to the line.

This approach yielded the four border lines in most cases. In case there were more than four lines, we removed those nearest to the center until there were only four lines. In case there were three lines, we searched for a single crossing on the open side (where there is no line), which is farthest away from the line on the opposite side. In case of success, the orientation of the new line was set to 180 minus the angle of the opposite line. With less than three lines, no further analysis was possible and the image analysis failed. If successful, pairwise crossings between lines were computed to get the corner points.

This complex approach worked for all but 3 (6%) of the 47 test images, and for all but one of the training images. Given that 12 (27.3%) of the remaining 44 test images were not perfectly classified (i.e. had at least one position wrong), these failures of the board estimation algorithm do contribute towards to the overall error rate. However, when analyzing video, estimation of board position

can be stretched over a longer period and thus can easily be made more stable, reducing this effect.

Initially, we interpolated linearily in two dimensions between the corner points to get the position of each field. However, this only works well if the board is relatively flat. Given known constraints on board size and aspect ratio, it was feasible to estimate the 3D position of each corner point by dynamic programming, and do a linear interpolation in 3D space which was then projected back into 2D space.

As final step, we recomputed all board lines where more than four crossings (according to the keypoint model) were within 16 pixels of the line. The result is a very good approximation of true board field positions, see also Figure 1, left image in middle row.

We are aware that this crucial step is only indirectly dependent on training data (by trial-and-error modifications of the algorithm) and as such is much harder to adapt to different tasks. Also, its performance is clearly below that of the other learning modules, but we found it quite hard to formulate this task as a learning problem. Still, we would like to point out that this step is the *only* step of our system that uses an ad-hoc algorithm.

4.4 Stone Detection

While the performance of the keypoint model concerning black and white stones was quite good, it was not perceived good enough for the given task, and there was also the problem that a nonnegligible proportion of stones did not yield any

keypoints at all and thus could not be detected via keypoint classification. We speculate that the uniform isotrop shape of the round stones makes them less significant for SIFT, and thus less likely to generate keypoints. For this specific task, (Loy & Zelinsky, 2003) propose a keypoint detector based on radial symmetry, which might have performed better. However, no useful implementation was readily available.

In initial experiments, we used a Hough transform for circles on an image preprocessed by the Sobel Edge Detector. However, as Sobel has no user-definable parameter to optimize, we switched to the state-of-the-art edge detector Canny and optimized its three parameters sigma, t_low and t_high on training data. Performance was measured by determining the optimal threshold on the proportion of border pixels within a circle around the known stone position, and using this to compute the number of erroneous classifications. The parameter set with the smallest number of errors was chosen.

We replaced the global Hough transform with a more efficient local variant, which only searches for circles in the vicinity of the estimated position ($\pm 1/3$rd of field size). Once the existence of a stone has been verified, the average brightness around its estimated central position is used to determine whether it is a black or a white stone, or an erroneous classification in case of intermediate brightness. Before computing average brightness, we applied a simple grayscale histogram equalization operator.

While the above works very well in most cases, there will always be times when a field type is not known after these steps. For this, we added two more

modules to get a definitive board field type for each field position: re-utilizing the keypoint classification model from 4.2 for stone and empty field detection, and a *Last Ditch Model* for the remaining unknown fields, using average and standard deviation of brightness of pixels in a window around the field position as features.

4.5 Keypoint Model for Stone Detection

If the keypoint classification model from Section 4.2. predicts a stone or a crossing near an unknown field position (less than half the local field size away), we take its result as definitive. This model is only applied to those fields without definitive outcome from any of the previous steps.

4.6 Last Ditch Model

In the case all above have failed, we still need a definitive outcome for all field positions. We have therefore trained a Logistic Regression model (le Cessie & van Houwelingen, 1992) on average and standard deviation of the brightness of all pixels in a circle around the field position, using the full training data of all field positions. The name *Last Ditch Model* was motivated by the fact that this is really our final attempt to find out what the field position contains, i.e. a last-ditch effort to work this out, which nevertheless is complementary to and independent of earlier approaches to ensure independent errors.

5 Results

We did two runs over test data: one with the linear SVM model for keypoint classification, and one with the optimized SVM model.

5.1 Optimized SVM

The best SVM for this task was a polynomial kernel with degree 3, switched-on feature-space normalization ($-F$) and a cost parameter $C = 10$. Of the 47 test images, 3 (6.38%) could not be analyzed because of problems in estimating the board position (Section 4.3). Of the 44 remaining images, 5 had minor mismatches in estimating the corner positions which we disregarded. Nevertheless, only 0.5±1.15 fields (out of 64) were misclassified on average per image. 72.7% of the images were recognized with no errors, the remaining had 1.83±1.59 errors. The optimized SVM learned its model in about one hour.

The 22 erroneously recognized field positions consisted of six errors due to the keypoint model (Section 4.5); six errors due to the Canny Stone Detector (4.4); and ten errors due to the last ditch model (4.6). The latter was used 113 times with an error rate of 8.84%, which is rather good for such a simple system, but still falls short of the other modules' performance.

The only disadvantage of this variant is that it is quite slow and takes about 114s per image, most of which is due to the optimized polynomial SVM.

Table 2 shows the results w.r.t. lighting conditions for the optimized polynomial kernel. As can be seen overall error is very small and relatively uniform, except for *Halogene Lamps*, which are by their nature spotlights and thus yield a

Lighting condition	Avg.Err	StD.Err	Avg.Keypoint	Avg.LastDitch
Bright Daylight	0.40	0.70	1.60	2.00
Cloudy Daylight	0.25	0.71	1.50	2.50
Halogene Lamps	1.44	2.13	3.11	4.11
Halogene Lamps & Daylight	0.22	0.44	0.67	2.00
Flourescent indirect	0.13	0.35	1.88	2.25

Table 2: Results for optimized SVM by lighting conditions: average and standard deviation of error, applications of keypoint model (Sec. 4.5) and LastDitch model (Sec. 4.6). Error are shown in field positions (e.g. 0.4 errors of 8*8 = 64 fields: 0.625% error)

radial pattern in brightness which makes the analysis harder. As support, note that flourescent indirect lighting, arguably the most uniform lighting condition, has the lowest error. We have also shown how often the last two steps (Sec.4.5 & 4.6) are invoked. Less than four out of sixty-four field positions per board depend on these two steps on average (except for *Halogene Lamps*), but they are nonetheless essential for good performance.

5.2 Linear SVM

For this, we used a linear kernel with $C = 1$. Of the 47 test images, 2 (4.25%) could not be analyzed because of problems in estimating the board position (Section 4.3). Of the 45 remaining images, 5 had minor mismatches in estimating the corner positions which we disregarded. 1.31±4.18 fields (out of 64) were misclassified on average per image. 62.2% of the images were recognized with

no errors, the remaining had 3.47±6.34 errors. All in all, the error rates are approximately doubled versus the optimized SVM. The linear SVM learned the model in about 7 minutes.

The 59 erroneously recognized field positions – almost three times the number from the optimized SVM – consisted of 13 errors due to the keypoint model (Section 4.5); 16 errors due to the Canny Stone Detector (4.4); and 30 errors due to the last ditch model (4.6). The latter was used 182 times with an error rate of 16.48%, which again approximately doubles the error rate versus the optimized SVM.

On the other hand, using the linear SVM reduces runtime to 8s per image, of which 2s are taken by the demo SIFT version and another 2s from an inefficient 3D calibrating routine, so a runtime of 5s would be easily achievable for the linear SVM by additional optimizations including using an optimized SIFT version.

Table 3 shows the results w.r.t. lighting conditions for the linear kernel. As can be seen overall error is quite non-uniform – *Cloudy Daylight* and *Flourescent indirect* perform much worse as with the optimized SVM. We speculate that the linear kernel is less able to compensate for different lighting conditions as reflected in the SIFT keypoint descriptors and thus the effects which are easily visible for the optimized kernel cannot be distinguished her.

5.3 Comparison to other approaches

To enable a comparison of our system to other multiple object recognition systems, we have adapted the trained linear and optimized SVM for usage as object

Lighting condition	Avg.Err	StD.Err	Avg.Keypoint	Avg.LastDitch
Bright Daylight	0.40	0.70	1.00	2.40
Cloudy Daylight	3.33	8.89	2.44	5.89
Halogene Lamps	1.33	2.00	1.78	4.89
Halogene Lamps & Daylight	0.13	0.35	0.25	2.38
Flourescent indirect	1.33	2.35	1.33	4.67

Table 3: Results for linear SVM by lighting conditions.

detectors.

Input were again the SIFT keypoint descriptors, and output was the class of each keypoint (crossing, t-segment, corner, white stone, black stone, and irrelevant). For clarity, we have only included one randomly chosen keypoint per board field, although there usually were several. In case there were no keypoints near board fields, we still counted these as legitimate objects and computed recall accordingly including these missing board fields. We also had to retrain both SVMs with an additional logistic model on the SVM outputs to get useful probability estimates which had a slight negative impact on performance. Note also that multi-class classification was done using the m-by-n methodology, i.e. training any class vs. any other class, thus yielding $\frac{6*5}{2} = 15$ binary SVM models.

Figure 4 shows Precision-Recall for the linear SVM, and Figure 5 shows the same for the optimized SVM. As you can see, optimizing the SVM has positive effects on the recall at same precision – most curves move to the left – but some classes have not improved (e.g. crossings and t-segments). It should be

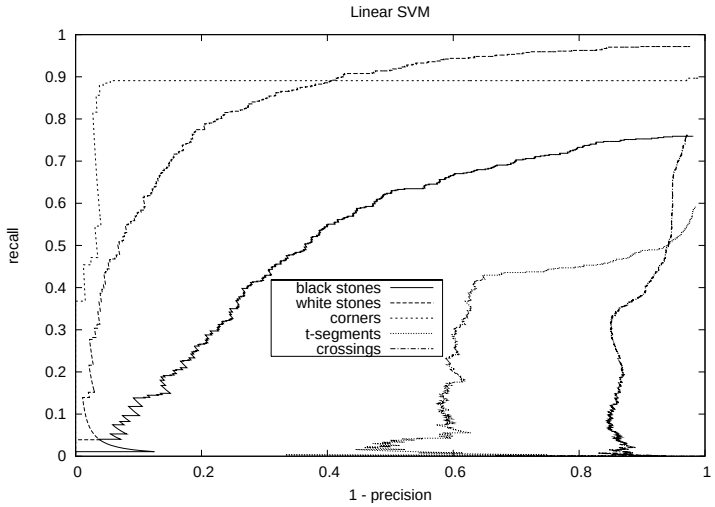


Figure 4: Precision-Recall curve for the linear SVM

obvious from these figures that without additional filtering, such as using two or more co-linear t-segments, the object recognition performance is inadequate for t-segments and crossings. However, for the other object classes the performance is quite good and resembles state-of-the-art systems. Note that the vast majority of keypoints (91.1%) were of type irrelevant (N), but both learning systems still managed to sort them to the bottom in the majority of classes. Especially the excellent performance for corners is quite surprising, as these were by far the smallest class in training data.

It would have been nice to compare the object recognition performance of this model with other recent systems for multiple object class detection such as (Mikolajczyk, Leibe & Schiele, 2006). However, we were unable to obtain an implementation of their work. Their quoted recognition times of 1-11s seem comparable versus our linear SVM.

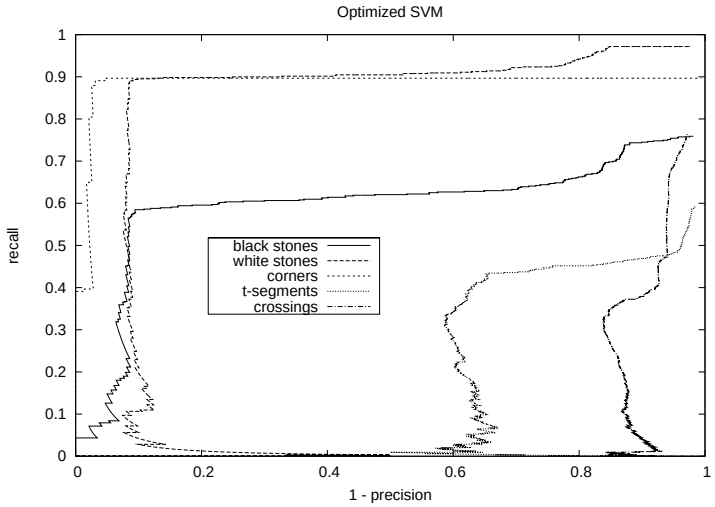


Figure 5: Precision-Recall curve for the optimized SVM

6 Real-Life performance

It was our intention to create a system robust w.r.t. lighting conditions, and we have largely reached this goal with the optimized SVM, albeit at a high computational cost. The linear kernel is much less robust concerning lighting conditions. We believe that the main reason for this robustness of the optimized kernel is the collection of training and test data roughly equally distributed over each lighting condition; the use of a non-linear kernel and parameter optimization on this training data; and the use of SIFT keypoint descriptors as input.

As we mentioned, the optimized SVM variant is able to recognize 72.7%, and the linear SVM variant 62.5%, of Go board images without error. This recognition accuracy could easily be further improved by recording several images, and taking the majority vote. Adding independent data on move positions,

e.g. from the sensor array of (Seewald, 2003), would also improve recognition accuracy.

We will now estimate the achievable recognition accuracy for the task of recording a full Go game from image samples of the game which are processed at maximum speed. For this, we must use the linear SVM variant. We assume an optimized version with 5s runtime per image (i.e. 0.2 frames-per-second). First, we note that Go is usually played on an 19x19 board, so we are basing this analysis on the per-field error which is independent of field size. The estimated probability of misclassifying a single field is thus equal to the per-field error: $prob_{field\_incorrect} = 0.0203$ (2.03%).

We assume a typical game to last 90 minutes, with 200 moves (100 for each player), and each move taking around one second during which the per-image accuracy is assumed to be 0% (i.e. no image is classified perfectly within this time period). This means that there are on average 26 seconds between moves. We assume that all errors are uncorrelated, so two correctly recognized field positions at the position of the move suffice for correctly recognizing the whole move. For a framerate *fps* of one-fifth frame per second (i.e. one frame every five seconds) this gives a move accuracy of $prob_{at\_least\_2\_correct} = 99.9998\%$, which translates into a probability of $prob_{whole\_game\_correct} = 99.96\%$ that the whole game is correctly recognized. For majority vote (at least 3 out of the 5 recognitions within 25s agree on the move) this would be $prob_{at\_least\_3\_correct} = 99.9919\%$ and $prob_{whole\_game\_correct} = 98.39\%$. The practical performance is likely to be higher, as there is a lot of redundancy between moves which this simple

probabilistic derivation does not account for. The following formulas allow to compute these probabilities for different values of k (i.e. desired minimum number of times the current move was correctly recognized), fps (fractional frame rate of recognition in frames per second), and $prob_{field\_incorrect}$ (probability that a single field is not recognized correctly). These formulas can also be used to determine the usefulness of similar systems with known move-recognition error rates and computation time to see whether their performance would be acceptable for this task.

$$\begin{aligned} prob_{at\_least\_k\_correct} &= \sum_{i=k}^{25fps} \binom{25fps}{i} (1 - prob_{field\_incorrect})^i \times \\ &\quad \times (prob_{field\_incorrect})^{25fps-i} \\ prob_{whole\_game\_correct} &= (prob_{at\_least\_k\_correct\_images})^{game\_length\_in\_moves} \end{aligned}$$

7 Discussion

Here, we aim to address some initial attempts to build the system and focus on what we tried that did not work.

Initially, we tested the Harris corner detector for Section 4.2, but its precision was far too low and it does not output meaningful region descriptors as SIFT, so the keypoint classification approach cannot be applied there. As is well-known, the SIFT keypoint detector is superior to earlier approaches, so this did not come as a surprise.

We also tested the Hough transform for lines initially for 4.3., but found out that for endgame positions, the board lines do not show clear enough patterns to be of use. If we were to analyze a video stream from a fixed camera beginning with the empty board, it would probably be feasible to use Hough transform to determine initial board position. As a complementary approach to the board estimation algorithm, this might show some promise, at least at the beginning of each game. However, to use it as such would probably reduce the robustness versus lighting conditions and camera position, which is why we refrained from following this further.

One final thing we have learned here is that it pays off to have several complementary modules which aim to solve the same task. A simple loop of analyzing errors on training data, choosing the module which has the highest contribution to overall error, and optimizing it, is very efficient in improving a mediocre system towards good performance. Overfitting should be avoided by using high bias learning algorithms, such as SVM with linear kernel and Logistic Regression, or by using a separate validation set.

8 Conclusion

We have presented a multi-strategical multi-object recognition system for a specific task, namely the recognition of final board position in the Japanese Game of Go. The system is able to correctly recognize between three thirds and three quarters of board positions perfectly under a large variety of lighting

conditions and camera positions. We claim the following contributions to the state-of-the-art for the presented system:

- High robustness versus changes in board position (auto-calibrating) and lighting conditions (all but *Halogene Lamps* show very small per-field errors of less than 0.4%), which has not been previously demonstrated for this task. However, this comes at high computational cost.

- Excellent performance sufficient for real-life application as Go game recorder, which has not be previously demonstrated for this task. Based on reasonable assumptions, already the faster and less robust linear SVM system could correctly record at least 98.39% of 19x19 Go games without any errors based on a real-time analysis of the video stream.

- Developed using a small training corpus. Only 100 images were created, half of them for training and half for evaluation – around 20 per lighting condition. Previously published approaches in multi-object recognition usually utilize much more training data for similar results.

- A practical application of diverse ensembles, where the different learning systems are complementary and applied in sequence, so that the ensemble is more accurate than any of its parts.

A exhaustive evaluation of robustness to lighting conditions versus design choices for training learning algorithms and parameter optimization would be useful, for example to investigate whether the same robustness can be achieved with less computational effort. However, this would need a formulation of the

board estimation task from Section 4.3 as a learning problem, which has so far eluded our grasp. It should also be noted that for this purpose, the amount of training data we have collected is probably not sufficient.

8.1 Acknowledgements

We would like to thank the author of SIFT for making a Linux version available for research purposes. Also, we would like to thank all people who were, however indirectly, involved in the original Intelligent Go Board project. Without their help and support, neither the IGO prototype nor this paper would exist. We would also like to thank the anonymous reviewers for their excellent feedback.

References

Ball C.J. 2004. Image2SGF project, `http://www.inference.phy.cam.ac.uk/cjb/image2sgf.html`. (accessed March 8, 2007)

Burger W., Burge M.J. 2006. Digitale Bildverarbeitung – Eine Einführung mit Java und ImageJ. Springer-Verlage Berlin Heidelberg. `http://www.imagingbook.com`

Canny J. 1986. A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6).

le Cessie S., van Houwelingen J.C. 1992. Ridge Estimators in Logistic Regression, Applied Statistics, 41(1):191–201.

Deuk-Cheol K., Ho-Joon K., Kyeong-Hoon J. 2005. Automatic Extraction of Game Record from TV Baduk Program. Proceedings of the 7th International Conference on Advanced Communication Technology (ICACT), (2):1185–1188.

Duda R.O., Hart P.E. 1972. Pattern Classification and Scene Analysis, J. Wiley, New York.

Harris C.G., Stephens M.J. 1988. A combined corner and edge detector, Proceedings of Fourth Alvey Vision Conference, Manchester, 1998, pp.147–151.

Hirsimäki, T. 2005. Extracting Go Game Positions from Photographs, Laboratory of Computer and Information Science, Helsinki University of Technology. `http://iki.fi/thirsima/gocam/` (accessed March 8, 2007)

Jacobs C.E., Finkelstein A., Salesin D.H. 1995. Fast Multiresolution Image Querying. Proceedings of SIGGRAPH 1995, ACM SIGGRAPH Annual Conference Series, pp.277–286.

LeCun Y., Bottou L., Bengio Y., Haffner P. 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-232.

Lowe D.G. 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60 (2), pp. 91-110.

Loy G., Zelinsky A. 2003. Fast Radial Symmetry for Detecting Points of Interest, IEEE Transactions on Pattern Analysis and Machine Intelligence.

Mikolajczyk K., Leibe B., Schiele B. 2006. Multiple Object Class Detection with a Generative Model, Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 26–36.

Platt J. 1998. Fast Training of Support Vector Machines using Sequential Minimal Optimization. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press.

Rasband, W.S. 1997-2006. ImageJ: Image Processing and Analysis in Java, U.S. National Institutes of Health, Bethesda, Maryland, USA. `http://rsb.info.nih.gov/ij/`

Sammon, J.W. 1969. A nonlinear mapping for data structure analysis. IEEE Transactions on Computers, C-18:401-409.

Scher, S. 2006. Hidden Markov Model Improves a Weak Classifier. Report, CS 290C, Winter 2006, taught by Prof. Manfred Warmuth. `http://www.soe.ucsc.edu/classes/cmps290c/Winter06/proj/stevenreport.doc` (accessed March 8, 2007)

Seewald A.K. 2003. The Intelligent Go Board project. `http://alex.seewald.at/IGO`.

Seewald A.K. 2005. Digits - A Dataset for Handwritten Digit Recognition. Technical Report, Österreichisches Forschungsinstitut für Artificial Intelligence, Wien, TR-2005-27.

Shiba K., Mori K. 2004. Detection of Go-board contour in real image using

genetic algorithm. In SICE 2004 Annual Conference, Vol.3, pp.2754–2759. ISBN 4-907764-22-7.

Witten I.H., Frank E. 2005. Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco. http://www.cs.waikato.ac.nz/~ml/weka